International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 4, Issue 1, Feb 2014, 7-16

USER SATISFACTION OVER QUERY RESULT RANKING IN WEB DATABASE SYSTEMS

PRAMOD KUMAR GHADEI[1] & S. SRIDHAR[2]

[1]Research Scholar, Sathyabama University, Chennai, Tamil Nadu, India

[2]Research Supervisor, Sathyabama University, Chennai, Tamil Nadu, India

ABSTRACT

To handle with the complication of enormous results returned from an online Web database in reply to a user asked query, in this paper we come up with a unique approach to rank the query results. Depend on the user query, we hypothesize how much the user bother about each and every attribute and assign a comparable weight to that. Then, all tuple in the query result, each attribute of every tuple is assigned with a score rate according to its "acceptable" to the user. For each tuple all attribute value scores are united as per the attribute weights to get a final ranking score for each tuple. Tuples with the leading ranking scores are presented to the user first as compare to other. This ranking method is independent of any domain and use feedback not required. Experimental results shows that this ranking method effectively satisfies users need with respect to his/her choices.

KEYWORDS: User Satisfaction, Query Result Ranking, Web Database, Attribute Value

INTRODUCTION

Due to fast development of the World Wide Web, progressively more new Web databases are available. Meanwhile, the size of existing Web databases is expanding promptly. Most common problem faced by all Web users is that there is *excessive query results* returned for an asked query. For example, when a user submits a query to sulekha.com to search for a rented house within Chennai city with a price range between ₹ 5,000 and ₹ 10,000, 6,856 records are returned. In order to catch "the best deal", for satisfaction the user has to go through this huge list and compare the houses among them, which is a laborious and time-consuming exercise.

Most of the Web databases rank their query results with respect to in ascending or descending order of a single attribute (e.g., sorted by date, sorted by price, etc.).On the other hand use satisfaction will met if multiple attributes are considered with respect to user choices. Although some extensions to SQL allow the user to define weights according to their own choices [21], [26], this approach is cumbersome and most likely hard to most users because they do not have clear idea how to set weights for different attributes. Moreover, the user assigned weight approach is not relevant for categorical attributes.

In this paper we handle the multiple query result issue by proposing an automated query ranking method, QRO (query ranking for online database), which can rank the queries for any online database without support of use feedback. We focus specifically on user satisfaction with respect to multiple queries in online databases. Most of the users do not have experience to express their choices to have a best deal. The houseDB Web database is used in the following examples to illustrate the inspiration on which QRO is established.

Example 1: Consider a rented house Web database W with a single table houseDB in which the house instances are stored as tuples with attributes: housetype, area, floor, rent, bedrooms, amenities and Location. Eachtuple r_i in W

represents a house for rent.

In tuple t_i and the query result R_q for a query q is submitted by a renter, we assign a ranking score to r_i, based on its attribute weights, which indicates its interest, from an E-business viewpoint, to the renter. For instance, it is obvious that a furnished, new and cheap house is globally popular and desired in the house renter market. However, sometimes the desired attribute values conflict with each other. For example, a new furnished house r with fewer amenities is unlikely to be cheap. Hence, we need to decide which attributes are more important for a renter. Some renter may care more about the house type, while some other buyer may be more concerned about its rent. For each attribute, we use a weight to denote its importance to the user.

Example 2: Given a query with condition "rent <5000", the query condition suggests that the user wants a relatively small house. Intuitively, besides the rent attribute, the user is more concerned about the amenities than he/she is concerned about the house_type and Location, considering that a relatively small house usually has low rent.

Given an unspecified attribute B_i, the correlation between B_i and the user query q is evaluated by the difference between the distribution of B_i's values over the query results and their distribution over the whole database W. The bigger the difference, the more B_i correlates to the specified attribute value(s). Consequently, we assign a bigger attribute weight to B_i.

Finally, the attribute weight and the value preference score are combined to get the final ranking score for each tuple. The tuples with the largest ranking scores are presented to the user first.

The contributions of this paper include the following:

- We present a unique approach to rank the tuples in the query results returned by online Web databases.
- We propose a new attribute importance learning approach, which is suitable to any domain and query adaptive.
- We also propose a new attribute-value choice score assignment approach for online Web databases.

In the entire ranking process, no user feedback is required.

The rest of the paper is organized as follows. Section 2 reviews some related work. Section 3 gives a formal definition of the many-query-result problem and presents an overview of QRO. Section 4 proposes our attribute importance learning approach while Section 5 presents our attribute preference score assignment approach. Experimental results are reported in Section 6. The paper is concluded in Section 7.

RELATED WORK

User Query result ranking has been measured in information retrieval for a long time ago. Cosine Similarity with TF-IDF weighting of the vector space model [3] and [18], the probabilistic ranking model [21] and [22] and the statistical language model [10] have been used successfully for ranking determination. In addition, [8], [9], [11] and [12] explore the integration of database and information retrieval techniques to rank tuples for text attributes. [2], [4] and [14] propose some keyword-query based retrieval techniques for databases. However, most of these techniques focus on text attributes and it is very difficult to apply these techniques to rank tuples with numerical or categorical attributes.

Few recent researches address the problem of relational query result ranking. In [7], [18], [20] and [24], user purpose feedback is employed to learn the similarity between a result record and the query, which is used to rank the query results in relational multimedia databases. In [21] and [18], the SQL query language is extended to allow the user to

specify the ranking function according to their preference for the attributes. In [18] and [15], users are required to building up profiles so that the query result is ranked according to their choice. Compared with the above work, our approach is fully automatic and does not need user feedback or any other human involvement.

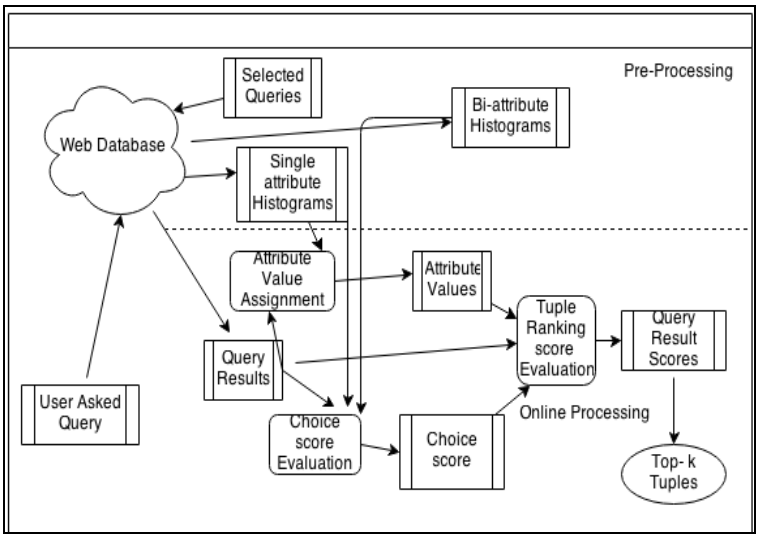

Figure 1: Architecture of a Web System for Implementing Query Result Ranking in Online Database (QRO)

USER QUERY RESULTS RANKING

We first define the user query-result problem and then present an overview of QRO.

Problem Formation

Deal with an autonomous Web database W with attributes $B=\{B_1, B_2, \ldots, B_n\}$ and a choice query q over D with a conjunctive choice condition of user that may include point queries, such as "$B_i = a_i$", or range queries, such as "$b_{i1} < B_i < b_{i2}$". Let $R=\{r_1, r_2, \ldots, r_n\}$ be the set of result tuples returned by W for the query q. In many cases, if q is not a selective query, it will produce a large number of query results (i.e., a large R). The goal is to develop a ranking function to rank the tuples in R that captures the user's choice, is domain-independent and does not require any type of user feedback.

QRO

Initially, we focus on online Web databases because online Web databases comprise a large portion of the databases on the Web. We further assume that each online Web database has a worth attribute, which we always assume to be B_1. The worth attribute B_1 plays an central role for all attributes during the attribute choice score assignment.

Table 1: Examples of Houses for Rent Tuples in Chennai City

	House Type	Area(Sqft)	Floor	Bedrooms	Amenities	Rent	Location
R1	Apartment	780	1st	2	Power, Security	7000	Adyar
R2	Independent	1300	2nd	3	Garden	95000	Tambram
R3	Service Apartment	1080	5th	2	Gym	11000	T Nagar
R4	Villa	2300	Ground	4	Swimming Pool	12000	Anna Nagar
R5	Gated Community	1200	6th	3	All	20000	Velachery

Having assigned a choicescore v_{ij} ($1 \leq j \leq m$) to each attribute-value of t_i and a weight w_j to the attribute B_j, the value choices scores v_{ij} are summed to obtain the ranking score c_i for t_i to reflect the importanceof attribute for the user. That is:

$$C_{i=}\sum_{j=1}^{n} w_j v_{ij}$$

The overall architecture of a system employing QRO is shown in Figure 1. Such a system includes two components: pre-processing component and online processing component. The pre-processing component collects statistics about the Web database *W* using a set of selected queries. Two types of histograms are built in the pre-processing step: single-attribute histograms and bi-attribute histograms. A single-attribute histogram is built for each attribute B_j. A bi-attribute histogram is built for each non-Price attribute (i.e., B_j in which $i>1$) using the Price attribute B_1.

Online-processing component ranks the query results given by the user query *q*. After getting all the query results *R* from the online Web database *W* for *q*, a value is assigned for each attribute by comparing its data distribution in *W* and in the query results *R*. At the same time, the choice score for each and every attribute value in the query result is determined using the information from the bi-attribute histograms. The attribute values and choices scores are combined to evaluate the ranking score for all tuple of the query result. The tuples' ranking scores are sorted and the top *K* tuples with the largest ranking scores are presented to the user earliest.

VALUEASSIGNMENT FOR TUPLE ATTRIBUTE

In a real world scenario, different types of users have various choices. Few people prefer deluxe house while some people care more about rent than anything else. Hence, we need to surmise the user's choice when we make recommendations to the user.. The difficulty of this problem lies in trying to determine what a user's choice is (i.e., which attributes are more important) when there is no user feedback.. To solve this problem, we start from the query the user submitted. We assume that the user's choice is reflected in the asked query and, hence, we use the query as a hint for assigning values to attributes.

Histogram Construction

To evaluate the KL-divergence in equation (1) we need to obtain the distribution of attribute values over *W*. The difficulty here is that we are dealing with an autonomous web database and we do not have full access to all the data. In the attribute value distribution over a collection of data crawled from W is used to estimate the real l attribute value distribution over *W*. the distribution of the crawled data can be different from that of *W* because the crawled data may be biased to the asked queries.

A histogram H_{Ti} also needs to be built for B_i over *R* (the result set) to get its probability distribution over *R*. For each bucket of H_{Di}, a bucket with the same bucket boundary is built in H_{Ti} while its frequency is counted in *R*.

Input: Attribute B_i and its value range Web database *W* with the total number of tuples | W | Minimum bucket number *n*

Output: A single-attribute histogram H_{Wi} for B_i

Method

1) If B_i is a categorical attribute
2) For each category b_{ij} of B_i, probe W using a query with condition "$B_i=b_{ij}$" and get its occurrence count c
3) Add a bucket (a_{ij}, c) into H_{wi}
4) If B_i is a numerical value attribute with value range $[b_{least}, b_{highest})$
5) $r = |W|/n$
6) less $= b_{least}$, high $= b_{highest}$
7) Do
8) probe W with a query with condition "low $=A_i<$high" and get its occurrence count c
9) if $c = r$
10) Add a bucket $(less, high, c)$ into H_{wi}
11) less = high, high $= b_{highest}$
12) else
13) high = less + (high - less) / 2
14) While less $< b_{highest}$
15) Return H_{wi}

Figure 2: Probing-Based Algorithm to Construct Histogram

Attribute Value Setting

After getting the histogram of B_i over W and R, the histograms are converted to a probability distribution by dividing the frequency in each and every bucket of the histogram by the bucket frequency sum of the histogram. That is, the probability distribution of B_i for W, P_{Wi},

The attribute value assignment is performed not only on the non worth attributes, but also on the worth attributes. If a worth attribute is a point condition, its attribute value will be the same for all tuples in the query result. If a specified attribute is a range condition, its attribute value t will be different for the tuples in the query result.

ATTRIBUTE CHOICE PREFERENCE SCORE ASSIGNMENT

Inclusion to the attributes themselves, various values of an attribute may have different types of attractions for the renter. For example, a house with a low rent is obviously more attractive than a high rent one if other attribute values are the same. Similarly, a house with low amenities is also obviously more desirable. Given an attribute value, the goal of the attribute choice score assignment module is to assign a choice score to it that reflects its desirableness for the renter. To facilitate the combination of scores of different attribute values, all scores assigned for different attribute values are needed.

Examples of Attribute Choice Score Assignment

Table 2 shows the average rent and assigned score for different type's house for the house DB database used in our experiments. It can be seen that the prices for different house types fit our intuition well. Deluxe house are evaluated to have a higher price than ordinary house and, consequently, are assigned a larger choice.

Table 2: House Type –Rent -Score Correspondence

House Types	Average Rent	Score
Apartment	5000	0.145
Individual House	9000	0.345
Serviced Apartment	12000	0.563
Villa	15000	0.765

EXPERIMENTAL SETUP

To evaluate how well different ranking approaches capture a user's choice, ten person were invited to participate in the experiments and behave as renter from the online web databases.

Databases

For our evaluation, we set up two databases from two domains in online service. The first database is a rented house database house DB (type, bedrooms, area, rent, amenities, Location) containing 15000 tuples extracted from sulekha.com. The attributes type, bed roomsl, area and Location are categorical attributes and the attributes rent and maintenance are numerical attributes. To simulate the Web databases for our experiments we used My SQL on a P5 3.2-GHz PC with 2GB of RAM. We implemented all algorithms in JAVA and connected to the RDBMS by ADO.

Implemented Algorithms

Besides QRO described above, we implemented two other ranking methods, which are described briefly below, to compare with QRO.

- RANDOM
- IRP(information retrieval using probability model)

Ranking Calculation

We now present a more formal evaluation of the query result ranking quality. A survey is conducted to show how well each ranking algorithm captures the user's choice. We evaluate the query results in two ways: average precision and user preference ranking.

Average Precision

In this experiment, each subject was asked to submit three queries for house DB and one query for house DB according to their preference. Each query had on average specified attributes for house DB and specified attributes for house DB. We found that all the attributes of house DB and house DB were specified in the collected queries at least once. On average, for house DB B, each query had a query result of 486 tuples, with the maximum being 3,213 tuples and the minimum 56 tuples. It can be seen that the many-query-result problem is a common problem in reality. Each query for house DB has a query result of 106 tuples on average.

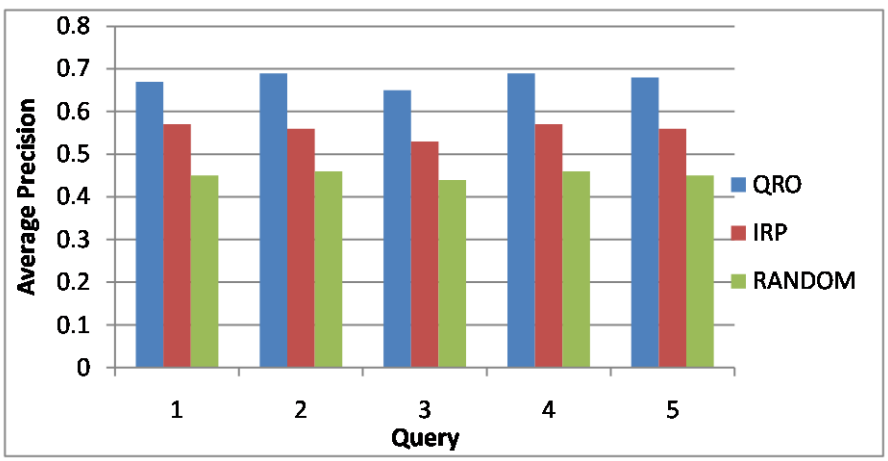

Figure 3: Average Precision for Different Ranking Methods for House DB

Performance Report

Using QRO, histograms need to be building up before the query results can be ranked. The histogram build up time depends on the number of buckets and the time to query the web to get the number of instance for each bucket. However, in most cases the histogram mostly does not change very much over time and so needs to be building up only once in a given time span.

Figure 3 shows the online execution time of the queries over houseDB as a function of the number of tuples in the query result. It can be seen that the execution time of QRO grows much linearly with the number of tuples in the query result. This is because ranking score sorting is fairly quick even for a large amount of data and thus most of the running time is spent in the first three modules.

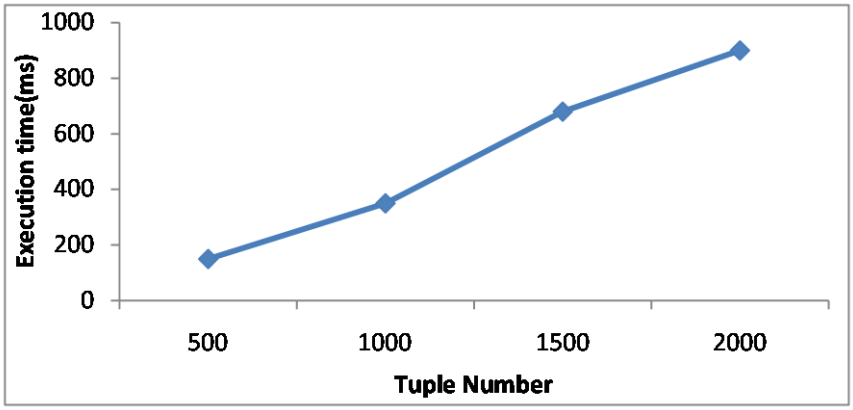

Figure 4: Execution Time for Different Numbers of Query Results

CONCLUSIONS

This paper presented, a unified automated ranking approach for the many user-query-result problem in online web database is proposed. Initiating from the user query, we assume that the key attributes are important for the user. We also assume that the attributes that are highly correlated with the query also are important to the user. We assign a value for each attribute according to its concern to the user. Then, for each value in each tuple of the query result, a choice score is assigned according to its desirableness in the online context, where users are assumed to more prefer services with lower prices. All choice scores are combined according to the attribute weight assigned to each attribute. No domain knowledge or user feedback is required in the whole system. Prior experimental results illustrate that QRO captures the user choices fairly well and better than earlier works.

ACKNOWLEDGEMENTS

We thank our institute for their support in research work progress. We greatly appreciate the reviewers and editor for their valuable suggestions and comments to improve this work.

REFERENCES

1. T.C. Zhou, H. Ma, M.R. Lyu, and I. King, “Userrec: A User Recommendation Framework in Social Tagging Systems,” Proc. 24th AAAI Conf. Artificial Intelligence, 2010.
2. Aboulnaga and S. Chaudhuri. “Self-tuning Histograms: Building Histograms Without Looking at Data,” *Proc. of the ACM SIGMOD Conf.*, 181-192, 1999.
3. S. Agrawal, S. Chaudhuri and G. Das. “DBXplorer: A System for Keyword Based Search over Relational

Databases," *Proc. of 18th Intl. Conf. on Data Engineering*, 5-16, 2002.

4. G. Bhalotia, C. Nakhe, A. Hulgeri, S. Chakrabarti and S. Sudarshan. "Keyword Searching and Browsing in Databases using BANKS," *Proc. of 18th Intl. Conf. on Data Engineering*, 431-440, 2002.
5. K. Chakrabarti, S. Chaudhuri and S. Hwang. "Automatic Categorization of Query Results," *Proc. of the ACM SIGMOD Conf.*, 755-766, 2004.
6. S. Chaudhuri, G. Das, V. Hristidis and G. Weikum. "Probabilistic Ranking of Database Query Results," *Proc. of the Intl. Conf. on Very Large Databases*, 888-899, 2004.
7. K. Chakrabarti, K. Porkaew and S. Mehrotra. "Efficient Query Refinement in Multimedia Databases," *Proc. of 16th Intl. Conf. on Data Engineering*, 196, 2000.
8. W. Cohen. "Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity," *Proc. of the ACM SIGMOD Conf.*, 201-212, 1998.
9. W. Cohen. "Providing Database-like Access to the Web Using Queries Based on Textual Similarity," *Proc. of the ACM SIGMOD Conf.*, 558-560,1998.
10. W.B. Croft and J. Lafferty. *Language Modeling for Information Retrieval*. Kluwer 2003.
11. N. Fuhr. "A Probabilistic Framework for Vague Queries and Imprecise Information in Databases," *Proc. of the 16 Intl. Conf. on Very Large Databases*th, 696-707, 1990.
12. N. Fuhr. "A Probabilistic Relational Model for the Integration of IR and Databases," *Proc. of the ACM SIGIR Conf.*, 309-317, 1993.
13. F. Geerts, H. Mannila and E. Terzi. "Relational Link-based Ranking," *Proc. of the 30th Intl. Conf. on Very Large Databases*, 552-563, 2004.
14. V. Hristidis and Y. Papakonstantinou. "DISCOVER: Keyword Search in Relational Databases," *Proc. of the 28th Intl. Conf. on Very Large Databases*, 670-681, 2002.
15. G. Koutrika and Y.E. Ioannidis. "Constrained Optimalities in Query Personalization," *Proc. of the ACM SIGMOD Conf.*, 73-84, 2005.
16. I. Muslea and T. Lee. "Online Query Relaxation via Bayesian Causal Structures Discovery," *Proc. of the AAAI Conf.*, 831-836, 2005.
17. U. Nambiar and S. Kambhampati. "Answering Imprecise Queries over Autonomous Web Databases," *Proc. of 22nd Intl. Conf. on Data Engineering*, 45, 2006.
18. M. Ortega-Binderberger, K. Chakrabarti and S. Mehrotra. "An Approach to Integrating Query Refinement in SQL," *Proc. Intl. Conf. on Extending Data Base Technology*, 15-33, 2002.
19. B. He, "Relevance Feedback," Encyclopedia of Database Systems,pp. 2378-2379, Springer, 2009.
20. Y. Rui, T.S. Huang and S. Merhotra. "Content-Based Image Retrieval with Relevance Feedback in MARS," *Proc. IEEE Intl. Conf. on Image Processing*, 815-818, 1997.
21. K. Sparck Jones, S. Walker and S.E. Robertson. "A Probabilistic Model of Information Retrieval: Development

and Comparative Experiments - Part 1," *Inf. Process. Management* **36**(6), 779-808, 2000.

22. K. Sparck Jones, S. Walker and S.E. Robertson. "A Probabilistic Model of Information Retrieval: Development and Comparative Experiments - Part 2," *Inf. Process. Manage*ment **36**(6), 809-840, 2000.
23. A. Jadhawar, A. S. Tamboli & S. S. Palkar" A Web Search Approach For Binary String In XML Data " (IJCSEITR) ISSN 2249-6831, TJPRC, Vol. 3, Issue 1, pp.123-132, Mar 2013,